09/361,641

	L #	Hits	Search Text	DBs	Time Stamp
1	L1	327341	recover$4	USPAT; US-PGPUB	2002/01/31 16:22
2	L2	55802	backup or back-up or (backing adj1 up)	USPAT; US-PGPUB	2002/01/31 16:22
3	L3	144108	restor$4	USPAT; US-PGPUB	2002/01/31 16:23
4	L4	8572	state$ adj1 information$	USPAT; US-PGPUB	2002/01/31 16:25
5	L5	1730	1 same 2	USPAT; US-PGPUB	2002/01/31 14:30
6	L6	233	5 same 3	USPAT; US-PGPUB	2002/01/31 14:30
7	L7	0	6 same 4	USPAT; US-PGPUB	2002/01/31 14:31
8	L8	34	6 and 4	USPAT; US-PGPUB	2002/01/31 14:32
9	L9	17643	data adj1 file$	USPAT; US-PGPUB	2002/01/31 14:35
10	L11	585	detect$4 with (state$ adj differen$4)	USPAT; US-PGPUB	2002/01/31 14:44
11	L12	2447	fail$4 adj4 computer$	USPAT; US-PGPUB	2002/01/31 14:46
12	L13	1	10 and 11	USPAT; US-PGPUB	2002/01/31 14:50
13	L14	4	10 and 12	USPAT; US-PGPUB	2002/01/31 15:02
14	L15	521	2 adj3 program$	USPAT; US-PGPUB	2002/01/31 15:05
15	L16	11	hard adj disk$ adj configurat$4	USPAT; US-PGPUB	2002/01/31 15:08
16	L17	449	3 adj3 computer$	USPAT; US-PGPUB	2002/01/31 15:10
17	L18	1	10 and 15	USPAT; US-PGPUB	2002/01/31 15:11
18	L19	0	10 and 16	USPAT; US-P PUB	2002/01/31 15:11
19	L20	2	10 and 17	USPAT; US-P PUB	2002/01/31 15:16

	L #	Hits	Search Text	DBs	Time Stamp
20	L21	604	2 adj2 c mputer$	USPAT; US-P PUB	2002/01/31 15:17
21	L22	25	21 with 1	USPAT; US-PGPUB	2002/01/31 15:18
22	L23	1	22 and framework	USPAT; US-PGPUB	2002/01/31 15:18
23	L10	15	8 and 9	USPAT; US-PGPUB	2002/01/31 15:31
24	L24	19	8 not 10	USPAT; US-PGPUB	2002/01/31 15:43
25	L25	2499	(714/?).ccls.	USPAT; US-PGPUB	2002/01/31 15:43
26	L26	50	6 and 25	USPAT; US-PGPUB	2002/01/31 15:43
27	L27	47	26 not 8	USPAT; US-PGPUB	2002/01/31 15:43
28	L28	293269	recover$4	EPO; JPO; DERWENT; IBM_TDB	2002/01/31 16:22
29	L29	36881	backup or back-up or (backing adj1 up)	EPO; JPO; DERWENT; IBM_TDB	2002/01/31 16:22
30	L30	80194	restor$4	EPO; JPO; DERWENT; IBM_TDB	2002/01/31 16:23
31	L31	6115	state$ adj1 information$	EPO; JPO; DERWENT; IBM_TDB	2002/01/31 16:25
32	L32	955	28 same 29	EPO; JPO; DERWENT; IBM_TDB	2002/01/31 16:25
33	L33	110	32 same 30	EPO; JPO; DERWENT; IBM_TDB	2002/01/31 16:26
34	L34	3	33 same 31	EPO; JPO; DERWENT; IBM_TDB	2002/01/31 16:26

DOCUMENT-IDENTIFIER: US 6240527 B1
TITLE: Method software and apparatus for saving using and recovering data

---------- KWIC ---------

BSPR:
Tape **backup** focuses on **backing up** an entire disk or specific files at a given moment in time. Typically the process will take a long time and is thus done infrequently (e.g., in the evening). Incremental **backups** involve only saving data that has changed since the last **backup**, thus reducing the amount of tape and **backup** time required. However, a fill system **recovery** requires that the initial full system **backup** and all subsequent incremental **backups** be read and combined in order to **restore** to the time of the last incremental **backup**.

BSPR:
A final example of why a user would want to revert to a backup is when the operating system gets corrupted (the executable or **data files** that are essential to run a computer) due, for example, to installing new software or device drivers that don't work.

BSPR:
The present invention is a method and apparatus for disk based information recovery in computer systems. This applies to all types of computer systems that utilize one or more hard disks (or equivalent), where the disks represent a non-volatile storage system or systems. Such types of computers may be, but are not limited to, personal computers, network servers, file servers, or mainframes. The invention stipulates using the otherwise unused pages or special dedicated pages on a hard disk in a circular fashion to store the recent original **states of information** on the disk that is altered. Collectively these extra pages represent a history buffer. These history pages can be intermixed with the OS's data and thus the present invention relies on re-mapping of disk locations between the OS and the actual hard disk. Using the information stored in the history buffer, another mapping can be

made
through which the state of the entire disk (excluding the extra pages) can be
reconstructed for any time in the past for as far back as the history buffer
contains information.

DEPR:
A computer's operating system (OS) typically stores information on a hard disk.
The example embodiments of the present invention present five fundamental
methods of recording the original **state of information** prior to its being
altered. The first four methods work substantially outside of the OS's method
of organizing and assigning its file to disk pages. They substantially differ
in performance and how they utilize the disk. The last method calls for
integrating the process of saving and retrieving original states of altered
information directly into the OS's filing system.

DEPR:
A basic purpose of the engine is to provide means for rolling back the state of
a disk to a previous time. This involves maintaining original and current
states and a mapping system to guide how these should be combined to create a
given state corresponding to some specific time in the past. In practice it is
not useful to restore a disk to a transitional **state where information** was in
the process of being updated. For example, if you were to save a word
processing document, you would like to see the disk either before or after a
save. Restoring to the time during the write process should be avoided since
there is no guarantee as to what the user would see. Therefore, the concept of
a safe point is introduced which corresponds to times at which the disk is
reasonably usable. These times are identified from large gaps in disk
activity, which are assumed to indicate the OS has flushed its caches, or
specific signals from the OS indicating such, when available.

DEPR:
When the OS overwrites data, the new data is placed in a CTMA block. Since the
new data is placed in unused pages in a CTMA block, diverting the writes here
inherently saves the overwritten **data, from the file's** viewpoint. How
this

saved (historic) data is tracked is discussed shortly. For now this description will focus on writing the new data.

DEPR:
In addition to supplying the data and the associated location key, the OS, when writing, can also supply a file identifier. If specified, this identifier allows the engine to direct new data from different files to different CTMA blocks. The engine allows a limited number of CTMA blocks to coexist in order to support the OS simultaneously writing to a limited number of files. By sending new **data for each file** to a different CTMA block, the engine de-fragments the files. As more CTMA blocks are supported at one time, the historic data is more rapidly discarded.

DEPR:
There are two basic overwrite situations. The first is that a small amount of **data in a file** is overwritten. In this case, assuming the file's existing allocation is optimal, it is best to swap the new data back in place while moving out the original state. On the other hand, if most of the file is overwritten, then it is best to leave the new data in its newly assigned locations, since these locations are likely optimal. The goal in both cases is reducing the amount of swapping. It is difficult to distinguish the cases at the time of the write since one cannot anticipate how much more data will be written in the future, and how fast (i.e., one could overwrite a file but over a long period of time). Further, if a file's size changes then leaving the new data where it is initially written likely reduces further re-arranging: If the size shrinks, then there will be space to recover (packing); if it increases, then perhaps separate areas will have to be combined.

DEPR:
All the methods thus far presented for saving original disk states are conceptually designed around a single disk. Of course, more than one disk may be involved, with their collective storage pooled into one large logical disk. The fault tolerance provided by the various methods deals with non-hardware failures like the user accidentally overwriting a file or a bug in an application corrupting files. However, there is also the case of the

disk
actually ceasing to function (i.e., if it broke and the information it contained is lost). **Recovery** from such a failure typically involves installing
a new hard disk, re-installing the operating system, and then **restoring** files
from a **backup** tape or similar device. This is a time-consuming process and
often involves some loss of data, that which was affected after the **backup**.

DEPR:
Thus a guaranteed usable backup image is available, and depending on the lag in
transferring changes, this point is likely not too far back in time. With a
RAID system, protection is achieved from a physical disk drive **failure, but**
none is provided for the computer crashing and leaving the last state of the
disk in transition.

DEPR:
There is a whole other category of **failures that occur in a computer** that have
nothing directly to do with the disk. They involve using an application over
an extended period of time during which information is manipulated in memory
and periodically (or at least at the session's conclusion) the information is
written to disk. A common failure results either from user errors or from bugs
in the applications, where something goes terribly wrong. So wrong, that in
fact, there is no easy way to recover. Any unsaved work is lost. Although
some applications try to minimize how much unsaved work is at risk (by automatic saves), it is still common for crashes to occur and for users to lose
a substantial time investment in unsaved work.

DEPR:
The prior paragraph discusses a new process for creating an incremental **backup**
tape. In truth, although the tape contains all the necessary information to
restore data from various points within a window of time, the organization of
the data on the tape is such that selective restoration (e.g., a single file)
is complicated. As a **backup** of a disk drive and its Driver's data, restoration
of the entire tape to a disk and the subsequent use of the normal Driver

software for **recovery** is the most natural and simplest means of accessing the
tape's data However, one may not always have an available disk drive to which
to **restore** the tape. Therefore, it is useful to include on the tape a directory that correlates the tape's data to their associated files, as written
at a certain time. Thus, when **restoring** data from tape, it is possible to
consult the directory to determine the portions of the tape that need to be
read. This pre-analysis allows the tape to be read in a single pass (assuming
the directory is at the front of the tape). The directory can map all the
various versions of files throughout the backed up window of tune, or just at
one time. In the latter case, the tape must be **restored** to disk in order to
access files across the window of time.

CLPR:
12. A method of accessing the prior **state of information** on a disk, comprising:

CLPR:
14. A method of accessing the prior **state of information** on a disk comprising,
keeping a record of the role of disk-based data elements X and Y, as managed by
an operating system and associated files, in order to preserve original data in
data element X;

CLPR:
16. A method of accessing the prior **state of information** on a disk, comprising:

CLPR:
17. A method of accessing the prior **state of information** on a disk as recited
in claim 16, further comprising:

CLPR:
18. A method of accessing the prior **state of information** on a disk as recited
in claim 17, wherein the releasing of oldest data frees disk space for other
uses.

CLPR:
19. A method of accessing the prior **state of information** on a disk as recited
in claim 16, further comprising:

CLPR:

20. A method of accessing the prior **state of information** on a disk, comprising:

CLPR:
21. A method of accessing the prior **state of information** on a disk as recited
in claim 20, further comprising:

CLPR:
22. A method of accessing the prior **state of information** on a disk as recited
in claim 21, wherein the releasing of oldest data frees disk space for other
uses.

CLPR:
23. A method of accessing the prior **state of information** on a disk as recited
in claim 20, further comprising:

DOCUMENT-IDENTIFIER: US 5987621 A
TITLE: Hardware and software failover services for a file server

---------- KWIC ---------

BSPR:
In accordance with a basic aspect of the invention, a network file server includes a cached disk array storage subsystem and a plurality of server computers linked to the cached disk array storage subsystem for responding to requests for file access from the clients in the data network by transferring data between the cached disk array storage subsystem and the data network. The server computers are programmed for executing tasks for responding to requests for file access from the clients in the data network. The server computers are also programmed for detecting a **failure of one of the server computers** causing interruption of execution of one of the tasks, and upon detecting the **failure, for causing another one of the server computers** to resume automatically the interrupted one of the tasks. In this fashion, the file server detects processor failure and resumes interrupted servicing of client requests with little or no client involvement.

BSPR:
In a preferred embodiment, **failure of a server computer** is detected by a failure to receive a signal that is normally transmitted by each of the server computers on a periodic basis. For example, the server computers are linked by a common data link over which each of the server computers transmits a "heartbeat" signal indicating the present condition of the server computer.

BSPR:
In the preferred embodiment, the server computers are programmed for executing tasks by maintaining **state information** of the tasks in memory of the cached disk array storage subsystem, and the server computers are programmed for resuming the interrupted tasks by recovering the **state information** for the interrupted tasks from the memory of the cached disk array, and resuming

execution of the interrupted tasks using the recovered **state information** for the interrupted tasks. The server computers are programmed for maintaining **state information** of the tasks in the memory of the cached disk array by writing updates of the **state information** to a log file in the memory of the cached disk array, and for recovering the **state information** of the interrupted ones of the tasks by reading from the log file updates of the **state information**. The tasks are subdivided into a series of transactions, each of the transactions includes a series of operations that can be repeated without causing substantial disruption during the resuming of execution of the interrupted tasks, and the server computers are programmed for maintaining **state information** of the tasks in memory of the cached disk array by committing results of each transaction to the memory of the cached disk array before committing results of a next transaction in the series of transactions.

BSPR:
In the preferred embodiment, the inactive controller server is programmed to respond automatically to a failure of the active controller server by resuming interrupted tasks of the active controller server, and becoming active in executing tasks for responding to requests for file access from clients in the data network by selecting respective ones of the stream server computers to service the requests. In particular, the active controller server is programmed for periodically sending a signal to the inactive controller server, and for executing tasks by maintaining **state information** of the tasks in memory of the cached disk array storage subsystem. The inactive controller server is programmed for responding to a failure to receive the signal by recovering the **state information** of the interrupted tasks from the memory of the cached disk array storage system, and becoming active in executing tasks for responding to requests for file access from clients in the data network. A flag, stored in the memory of the cached disk array, indicates which of the controller servers is active and which of the controller servers is inactive. The controller servers are programmed to read the flag to determine whether or not to become

active in executing tasks for responding to requests for file access from clients in the data network. The inactive controller server is programmed to become active in response to a failure to receive the signal by changing the flag so that the active controller server becomes inactive. The active one of the controller servers is programmed to read the flag before changing the **state information** of the tasks in the memory of the cached disk array storage subsystem, so that the active controller server will not change the **state information** of the tasks in the memory of the cached disk array a certain time after the inactive controller server becomes active.

BSPR:
In the preferred embodiment, the active controller server is programmed to perform a stream server failover task for detecting a **failure of one of the stream server computers** and in response selecting another one of the stream server computers to resume servicing of a client request interrupted by the **failure of the stream server computer**. When the active controller server fails to receive a "heartbeat" signal from a stream server computer, the active controller server checks whether or not the failed stream server is currently servicing any client requests. If so, the active controller server checks a failover mode selection signal included in the client request. The failover mode selection signal indicates whether or not the active controller server should operate in a transparent failover mode or in a client control failover mode. In the transparent failover mode, the active controller server re-routes a data stream from the stream server computer selected to resume servicing of the client request. In the client control failover mode, the active controller server informs the client of the stream server computer failure, and the client re-routes a data stream from the stream server computer selected to resume servicing of the client request. Re-routing of the data stream, for example, involves the active controller server or the client re-programming an address decoder in a network switch coupled to the steam server computers. It is

possible that neither the active controller server nor the client has the capability of re-routing the data stream. For this reason, the controller server is programmed to default to the client control failover mode when the client has selected the transparent failover mode and the client but not the controller server has the capability for re-routing a data stream from the stream server computer selected to resume servicing of the client request.

DEPR:
In a preferred mode of operation, to archive **data from a file** from the network to tape, one of the stream servers 21 receives the file from the network 25 and prestages the file to the cached disk array 23 at a high rate limited by the network transmission rate (about 150 GB/hour). Then one of the stream servers 21 destages the file from the cached disk array 23 to an associated one of the read/write stations 51 at a tape device speed (about 7 GB/hour). For most applications, prestaging to disk can be done immediately, and staging from disk to tape including sorting of files onto respective tape cassettes can be done as a background operation or at night, when the load on the video server is at a minimum. In this fashion, the cached disk array 23 can absorb a high data inflow aggregation from tens or hundreds of network links streaming from multiple sites, and balance this load on the read/write stations 41. Prestaging to the cached disk array allows better use of the read/write stations 51, matching of server flow to tape streaming flow, and reduction of tape and read/write station wear. Prestaging to the back-end also allows multiple classes of backup and restore services, including instant backup for files maintained on disk in the cached disk array, and temporary batch backup pending a success or failure acknowledgment. Prestaging to the cached disk array 23 also makes economical an on-line archive service performing the staging from the cached disk array to tape as a background process.

DEPR:
In the processors of controller servers 28, 29, a software application is run by a general purpose operating system such as Microsoft NT, and a

network
client communicates service requests to the video file server only through the
software application executing on an active one of the controller servers 28,
29. This software application executes as a central control to prevent the
video file server from performing conflicting operations in response to
concurrent requests from various network clients. For example, the video file
server should not erase a file for one client while **data from the file** is being
streamed to another client.

DEPR:
Turning now to FIG. 19, there is shown a schematic diagram illustrating the
flow of data through the file server (20 in FIG. 1) in a "network backup"
operation. The stream servers 21 serve to funnel data from clients on the
network 25 into the cached disk array 23. The stream servers accept data at a
rate on the order of, typically, several megabits per second from each network
client (as determined by the existing network connections and remote backup
application capabilities). Each stream server sends data to the cached disk
array 23 at a rate which is the aggregate of all the streams received by the
stream server and can be on the order of about fifty to one hundred megabits
per second. The cached disk array in turn sends the backup data to the tape
silo 24 at the rate allowed by the capabilities of the tape silo--typically on
the order of 2 to 20 megabits per second, much less than the capabilities of
the cached disk array. (Disk arrays are typically faster than tape silos, as
determined by the ratio of the concurrent number of disk read/write streams to
the number of tape read/write drives.) The cached disk array serves as a speed
matching buffer and as a means for combining the **data or files** to be written to
a particular tape cartridge in the tape silo 24. Backup data can be streamed,
concurrently by all the stream servers, to the cached disk array 23 at an
aggregate speed on the order of 150 gigabytes per hour. The backup data are
then sorted and written to tape at a tape transport speed on the order of 7
gigabytes per hour per device.

DEPR:
Because the cached disk array 23 may use a nonvolatile write buffer and well-known RAID techniques of error correction to **recover** from disk drive failures, the cached disk array can acknowledge completion of a **backup** operation as soon as the data are written to the cached disk array. The actual writing to tape could be done as a background process, mainly during off-peak hours, when the stream servers are not heavily loaded by data transfers to and from network clients. The cached disk array can provide "instant" **restore** service for **backup** files maintained in the cached disk array. The cached disk array can also provide temporary batch **backup,** without writing to tape, pending success or failure of transactions by clients that employ transactional semantics or transaction processing.

DEPR:
The subroutine depicted as a flowchart in FIGS. 23 and 24 uses the stripe set identification number for specification of locations for accessing desired continuous media data. In a data network, however, it is conventional for network clients to specify desired **data by a file** name. Preferably, continuous media data is logically organized as individually named files, which will be referred to as "clips".

DEPR:
As shown in FIG. 2, the video file server 20 has dual redundant controller servers 28, 29 and multiple stream servers 21 to permit recovery from controller server and stream server failures. Preferably, the recovery is performed with little or no disruption of client services and little or no client involvement. The programming in the video file server 20 that coordinates the recovery operations is referred to as failover services. A major portion of this programming is included in the controller server software. In general, controller server or stream server failover involves recovery of **state information** necessary for resuming the interrupted operations of the failed controller server or stream server, and the resumption of the interrupted operations by the alternative controller server or an alternative stream server.

DEPR:
As shown in FIG. 2, the **state information** necessary for resuming the interrupted operations of the failed controller server is maintained in the cached disk array 23 and normally resides at least in the cache memory 41 of the cached disk array 23. This **state information** includes "controller server ephemeral atomic state" 501, and may also include **state information** in one of the log files 502.

CLPR:
2. The file server as claimed in claim 1, wherein each of the server computers is programmed for periodically sending a signal to the other server computers, and wherein the other server computers are programmed to resume a task of said each of the server computers in response to a **failure of said each of the server computers** to periodically send the signal.

CLPR:
4. The file server as claimed in claim 1, wherein the plurality of server computers are programmed for executing tasks by maintaining **state information** of the tasks in memory of the cached disk array storage subsystem, and wherein the server computers are programmed for resuming interrupted ones of the tasks by recovering the **state information** for the interrupted ones of the tasks from the memory of the cached disk array, and resuming execution of the interrupted ones of the tasks using the recovered **state information** for the interrupted ones of the tasks.

CLPR:
5. The file server as claimed in claim 4, wherein the plurality of server computers are programmed for maintaining **state information** of the tasks in the memory of the cached disk array by writing updates of the **state information** to a log file in the memory of the cached disk array, and for recovering the **state information** of the interrupted ones of the tasks by reading from the log file updates of the **state information**.

CLPR:
6. The file server as claimed in claim 4, wherein the tasks are

subdivided
into a series of transactions, each of the transactions includes a series of
operations that can be repeated without causing substantial disruption during
the resuming of execution of the interrupted ones of the tasks, and the server
computers are programmed for maintaining **state information** of the tasks in
memory of the cached disk array by committing results of each transaction to
the memory of the cached disk array before committing results of a next
transaction in the series of transactions.

CLPR:
9. The file server as claimed in claim 8, wherein the memory of the cached
disk array stores system **state information** indicating which of the controller
servers is active and which of the controller servers is inactive, the
controller servers are programmed to read the system **state information** to
determine whether or not to become active in executing tasks for responding to
requests for file access from clients in the data network by selecting
respective ones of the stream server computers to service the requests, and
wherein the inactive one of the controller servers is programmed to become
active in response to a failure to receive the signal by changing the system
state information so that the active one of the controller servers becomes
inactive.

CLPR:
10. The file server as claimed in claim 9, wherein the active one of the
controller servers is programmed to read the system **state information** before
changing the **state information** of the tasks in the memory of the cached disk
array storage subsystem, so that the active one of the controller servers will
not change the **state information** of the tasks in the memory of the cached disk
array a certain time after the inactive one of the controller servers becomes
active.

CLPR:
11. The file server as claimed in claim 8, wherein the plurality of server
computers are programmed for maintaining **state information** of the tasks in the
memory of the cached disk array by writing updates of the **state**

information to
a log file in the memory of the cached disk array, and for recovering the **state**
information of the interrupted ones of the tasks by reading from the log file
updates of the **state information**.

CLPR:
12. The file server as claimed in claim 8, wherein the tasks are subdivided
into a series of transactions, each of the transactions includes a series of
operations that can be repeated without causing substantial disruption during
the resuming of execution of the interrupted ones of the tasks, and the
controller servers are programmed for maintaining **state information** of the
tasks in memory of the cached disk array by committing results of each
transaction to the memory of the cached disk array before committing results of
a next transaction in the series of transactions.

CLPR:
13. The file server as claimed in claim 7, wherein the active one of the
controller servers is programmed to perform a stream server failover task for
detecting a **failure of one of the stream server computers** and in response
selecting another one of the stream server computers to resume servicing of a
client request interrupted by the **failure of one of the stream server computers**.

CLPR:
14. The file server as claimed in claim 13, wherein the controller servers and
the stream server computers share a data link over which are periodically
transmitted status signals including a status signal transmitted from the
active one of the controller servers to the inactive one of the controller
servers and a status signal transmitted from each of the stream server
computers to the active one of the controller servers, and wherein the inactive
one of the controller servers is programmed for detecting a failure of the
active one of the controller servers upon failing to receive the status signal
from the active one of the controller servers, and wherein the active one of
the controller servers is programmed for detecting a **failure of one of the**
stream server computers upon failing to receive the status signal from one of

the stream server computers.

CLPR:
18. The file server as claimed in claim 15, wherein the client request includes a failover mode selection signal, and wherein the controller server is programmed for checking the failover mode selection signal to select one of a plurality of predetermined modes for resuming servicing of the client request found to have been interrupted by the **failure of one of the stream server computers**.

CLPR:
25. The method as claimed in claim 24, wherein each of the server computers periodically sends a signal to the other server computers, and wherein one of the other server computers resumes a task of said each of the server computers in response to a **failure of said each of the server computers** to periodically send the signal.

CLPR:
26. The method as claimed in claim 24, wherein the plurality of server computers execute tasks by maintaining **state information** of the tasks in memory of the cached disk array storage subsystem, and wherein the server computers resume interrupted ones of the tasks by recovering the **state information** for the interrupted ones of the tasks from the memory of the cached disk array, and resuming execution of the interrupted ones of the tasks using the recovered **state information** for the interrupted ones of the tasks.

CLPR:
27. The method as claimed in claim 26, wherein the plurality of server computers maintain **state information** of the tasks in the memory of the cached disk array by writing updates of the **state information** to a log file in the memory of the cached disk array, and recovering the **state information** of the interrupted ones of the tasks by reading from the log file updates of the **state information**.

CLPR:
28. The method as claimed in claim 26, wherein the tasks are subdivided into a series of transactions, each of the transactions includes a series of

operations that can be repeated without causing substantial disruption during
the resuming of execution of the interrupted ones of the tasks, and the controller servers maintain **state information** of the tasks in memory of the
cached disk array by committing results of each transaction to the memory of
the cached disk array before committing results of a next transaction in the
series of transactions.

CLPR:
31. The method as claimed in claim 30, wherein the memory of the cached disk
array stores system **state information** indicating which of the controller
servers is selected to be active and which of the controller servers is selected to be inactive, the controller servers read the system **state information** to determine whether or not to become active in executing tasks for
responding to requests for file access from clients in the data network by
selecting respective ones of the stream server computers to service the requests, and wherein the inactive one of the controller servers becomes active
in response to a failure to receive the signal by changing the system **state**
information so that the active one of the controller servers becomes inactive.

CLPR:
32. The method as claimed in claim 31, wherein the active one of the controller servers reads the system **state information** before changing the **state**
information of the tasks in the memory of the cached disk array storage subsystem, so that the active one of the controller servers will not change the
state information of the tasks in the memory of the cached disk array a certain
time after the inactive one of the controller servers becomes active.

CLPR:
33. The method as claimed in claim 30, wherein the controller servers maintain
state information of the tasks in the memory of the cached disk array by
writing updates of the **state information** to a log file in the memory of the
cached disk array, and recovering the **state information** of the interrupted ones
of the tasks by reading from the log file updates of the **state information**.

CLPR:
34. The method as claimed in claim 30, wherein the tasks are

subdivided into a
series of transactions, each of the transactions includes a series of operations that can be repeated without causing substantial disruption during
the resuming of execution of the interrupted ones of the tasks, and the controller servers are programmed for maintaining **state information** of the
tasks in memory of the cached disk array by committing results of each transaction to the memory of the cached disk array before committing results of
a next transaction in the series of transactions.

CLPR:
35. The method as claimed in claim 29, wherein the active one of the controller servers performs a stream server failover task for detecting a
failure of one of the stream server computers and in response selecting another
one of the stream server computers to resume servicing of a client request
interrupted by the **failure of one of the stream server computers**.

CLPR:
36. The method as claimed in claim 35, wherein the controller servers and the
stream server computers share a data link over which are periodically transmitted status signals including a status signal transmitted from the
active one of the controller servers to the inactive one of the controller
servers and a status signal transmitted from each of the stream server computers to the active one of the controller servers, and wherein the inactive
one of the controller servers detects a failure of the active one of the
controller servers upon failing to receive the status signal from the active
one of the controller servers, and wherein the active one of the controller
servers detects a **failure of one of the stream server computers** upon failing to
receive the status signal from one of the stream server computers.

CLPR:
40. The method as claimed in claim 37, wherein the client request includes a
failover mode selection signal, and wherein the controller server checks the
failover mode selection signal to select one of a plurality of predetermined
modes for resuming servicing of the client request found to have been interrupted by the **failure of one of the stream server computers**.

CLPV:
wherein the interrupted one of the tasks includes a series of transactions, and

said plurality of server computers are programmed to record **state information** for the interrupted one of the tasks during execution of the interrupted one of the tasks, and for resuming automatically the interrupted one of the tasks by recovering the recorded **state information** for the interrupted one of the tasks including **state information** from at least one transaction completed prior to the interruption of the interrupted one of the tasks, and resuming execution of a next one of the transactions in the interrupted one of the tasks using the recovered **state information** from said at least one of the transactions completed prior to the interruption of the interrupted one of the tasks.

CLPV:
wherein an interrupted one of said tasks includes a series of transactions, and the controller servers are programmed to record **state information** for the interrupted one of the tasks during execution of the interrupted one of the tasks, and for resuming automatically the interrupted one of the tasks by recovering the recorded **state information** for the interrupted one of the tasks including **state information** from at least one transaction completed prior to the interruption of the interrupted one of the tasks, and resuming execution of a next one of the transactions in the interrupted one of the tasks using the recovered **state information** from said at least one of the transactions completed prior to the interruption of the interrupted one of the tasks.

CLPV:
the active one of the controller servers is programmed for periodically sending a signal to the inactive one of the controller servers, and for executing tasks by maintaining **state information** of the tasks in memory of the cached disk array storage subsystem, and

CLPV:
the inactive one of the controller servers is programmed for responding to a failure to receive the signal by recovering the **state information** of the interrupted tasks from the memory of the cached disk array storage system, and becoming active in executing tasks for responding to requests for file

access
from clients in the data network by selecting respective ones of the stream
server computers to service the requests.

CLPV:
wherein the controller server is programmed to perform a stream server failover
task for detecting a **failure of one of the stream server computers** and in
response selecting another one of the stream server computers to resume
servicing of a client request interrupted by the **failure of one of the stream
server computers**;

CLPV:
wherein the stream servers are programmed to execute tasks to service client
requests, and an interrupted one of the tasks includes a series of
transactions, and said stream server computers are programmed to record **state
information** for the interrupted one of the tasks during execution of the
interrupted one of the tasks, and upon interruption of the interrupted one of
the tasks, for resuming automatically the interrupted one of the tasks by
recovering the recorded **state information** for the interrupted one of the tasks
including **state information** from at least one transaction completed prior to
the interruption of the interrupted one of the tasks, and resuming execution of
a next one of the transactions in the interrupted one of the tasks by using the
recovered **state information** from said at least one of the transactions
completed prior to the interruption of the interrupted one of the tasks.

CLPV:
wherein the file server is programmed to perform stream server failover, in
response to a **failure of said one of the stream server computers,** in which
another one of the stream server computers resumes servicing of a client
request interrupted by the **failure of said one of the stream server computers,**
and

CLPV:
wherein the client request includes a failover mode selection signal, and
wherein the file server is programmed for checking the failover mode selection

signal to select one of a plurality of predetermined modes for resuming servicing of the client request found to have been interrupted by the **failure of said one of the stream server computers**.

CLPV:
(b) detecting a **failure of one of the server computers** causing interruption of execution of one of the tasks, and upon detecting the **failure, causing another one of the server computers** to resume automatically the interrupted one of the tasks;

CLPV:
wherein the interrupted one of the tasks includes a series of transactions, execution of the interrupted one of the tasks includes recording **state information** from at least one transaction completed prior to the interruption of the interrupted one of the tasks, and resuming automatically the interrupted one of the tasks includes recovering the recorded **state information** from said at least one transaction completed prior to the interruption of the interrupted one of the tasks, and resuming execution of a next one of the transactions in the interrupted one of the tasks using the recovered **state information** from said at least one of the transactions completed prior to the interruption of the interrupted one of the tasks.

CLPV:
wherein an interrupted one of said tasks includes a series of transactions, and the active one of the controller servers records **state information** for the interrupted one of the tasks during execution of the interrupted one of the tasks, and the inactive one of the controller servers resumes automatically the interrupted one of the tasks by recovering the recorded **state information** for the interrupted one of the tasks including **state information** from at least one transaction completed prior to the interruption of the interrupted one of the tasks, and resuming execution of a next one of the transactions in the interrupted one of the tasks using the recovered **state information** from said at least one of the transactions completed prior to the interruption of the interrupted one of the tasks.

CLPV:
the active one of the controller servers periodically sends a signal to the
inactive one of the controller servers, and executes tasks by maintaining **state information** of the tasks in memory of the cached disk array storage subsystem,
and

CLPV:
the inactive one of the controller servers responds to a failure to receive the
signal by recovering the **state information** of the interrupted tasks from the
memory of the cached disk array storage system, and becoming active in executing tasks for responding to requests for file access from clients in the
data network by selecting respective ones of the stream server computers to
service the requests.

CLPV:
the controller server performing a stream server failover task including the
detection of a **failure of one of the stream server computers,** and in response
selecting another one of the stream server computers to resume servicing of a
client request interrupted by the **failure of one of the stream server computers;** and

CLPV:
said another one of the stream server computers resuming the servicing of the
client request having been interrupted by the **failure of one of the stream server computers;**

CLPV:
wherein the servicing of the client request interrupted by the **failure of one of the stream server computers** includes execution of an interrupted task
including a series of transactions, execution of the interrupted task includes
recording **state information** from at least one transaction completed prior to
the interruption of the interrupted task, and the resuming of servicing of the
client request having been interrupted by the **failure of one of the stream server computers** includes recovering the recorded **state information** from said
at least one transaction completed prior to the interruption of the

interrupted
task, and resuming execution of a next one of the transactions in the interrupted task using the recovered **state information** from said at least one
of the transactions completed prior to the interruption of the interrupted
task.

CLPV:
the file server performing stream server failover in response to a **failure of**
said one of the stream server computers in which another one of the stream
server computers resumes servicing of a client request interrupted by the
failure of said one of the stream server computers;

CLPV:
wherein the client request includes a failover mode selection signal, and
wherein the file server checks the failover mode selection signal to select one
of a plurality of predetermined modes for resuming servicing of the client
request found to have been interrupted by the **failure of said one of the stream**
server computers.

CLPW:
(b) detecting a **failure of one of the server computers** causing interruption of
execution of one of the tasks, and upon detecting the **failure, for causing**
another one of the server computers to resume automatically the interrupted one
of the tasks;